(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 March 2002 (14.03.2002)

PCT

(10) International Publication Number
WO 02/21281 A2

(51) International Patent Classification[7]: G06F 11/32

(21) International Application Number: PCT/US01/29049

(22) International Filing Date:
10 September 2001 (10.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/658,208 8 September 2000 (08.09.2000) US

(71) Applicant: NETWORK APPLIANCE, INC. [US/US]; 495 East Java Drive, Sunnyvale, CA 94089 (US).

(72) Inventor: BAGG, Roderick E.; 1157 Shasta Avenue, San Jose, CA 95126 (US).

(74) Agent: SWERNOFSKY, Steven, A.; Swernofsky Law Group, P.O. Box 390013, Moutain View, CA 94039-0013 (US).

(81) Designated States *(national)*: CA, JP.

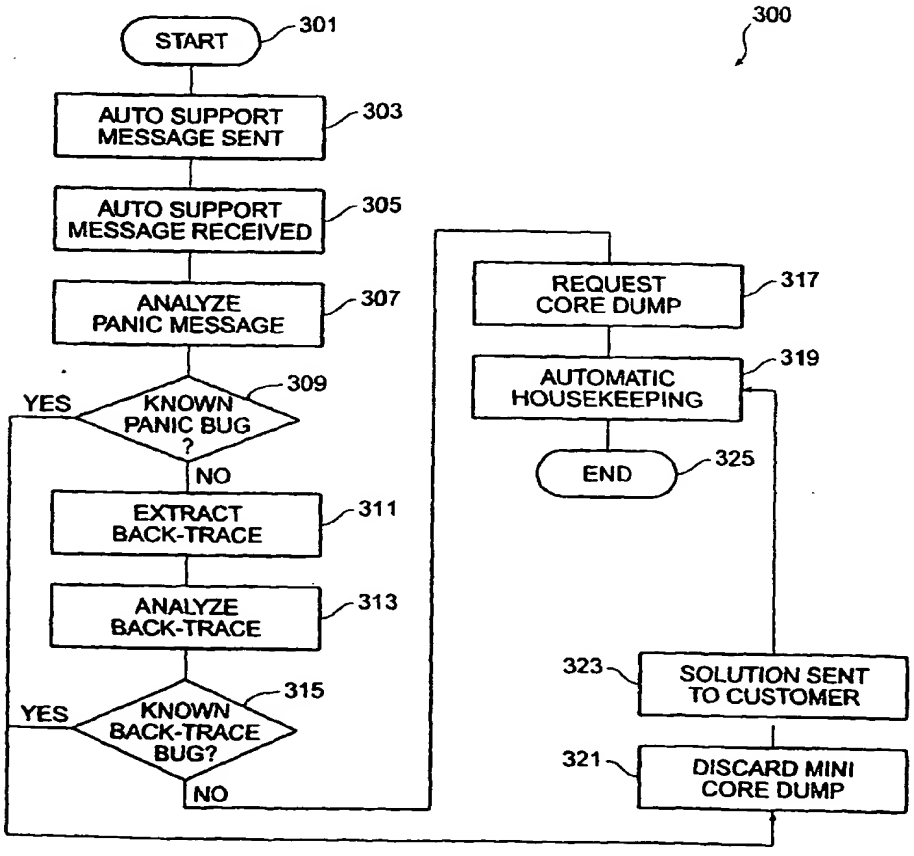
(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: PANIC MESSAGE ANALYZER



(57) Abstract: The invention provides a method and system for automatically obtaining and analyzing error messages from end users of software and hardware products. Additionally, the invention provides a method and system of providing solutions that automatically and manually correct the errors. An ever-growing database of errors and solutions is maintained so that future identical or similar problems are expedited without human intervention. Successful analysis at any but the lowest level automatically allows previous levels to be taught greater efficiency for future analysis of the same or similar errors.

WO 02/21281 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

PANIC MESSAGE ANALYZER

Background of the Invention

1. *Field of the invention*

This invention relates to analysis of panic messages from network servers.

2. *Related Art*

Computers rely on programmed instructions to direct their operation. General purpose computers most often receive these instructions from software that executes within their memory. Despite the fact that software engineers vigorously test programs to eliminate the presence of coded instructions that may cause errors (commonly known as bugs), the presence of bugs is practically unavoidable in simple programs and a foregone conclusion in complex programs.

When a computer executes a program instruction that causes an error, the error may have relatively no effect on the system, or the error may cause the system to crash. All errors are of importance to software engineers, however, those that cause a catastrophic result, such as a crash, are of primary importance. Generally, systems are designed to provide an alert to a system operator that they have suffered some type of failure. Error messages provide this alert, and these messages are useful when reporting errors to a manufacturer of the software.

A first known method to enable reporting of a software application error is to provide a pre-public release of a software package to a select group customers for "beta testing." During this trial period, customers report to the company any problems that they encounter and the software engineers at the company fix the bugs and provide updated versions of the software to the beta testers who continue testing with the new version. This process continues for a short testing period until the software is hopefully error free.

While this first known method provides reporting of software bugs to a manufacturer it suffers from several drawbacks. First, it provides no method for automatically reporting the problem to the manufacturer. It relies solely on the beta tester to inform the manufacturer. Second, it provides no automated analysis of a problem identified by a beta tester. That is, it requires an employee at the manufacturer to determine whether the problem has already been reported, fixed, or is a new problem. Third, it provides no method for delivery of updated software to a user who is determined to be using older software with an identified and fixed problem.

A second known method of reporting computer system errors is to rely on the end user to call the manufacturer and report a problem when it occurs. The customer is provided a customer support line that they may call to report problems they are having. Based on the frequency and subject matter of calls received, the manufacturer may conclude there is a problem with some portion of a program.

While this second known method provides reporting of software bugs to a manufacturer it suffers from several drawbacks. First, the customer may decide not to call as customer support calls tend to involve long waits on hold listening to musak and often provides no relief as the manufacturer has no formal structure in place to coordinate and analyze the calls they receive. Second, the customer may not be knowledgeable enough to provide the manufacturer with the necessary information they need to diagnose the problem, or worse, they may misinform the manufacturer as to the origin of the problem.

Accordingly, it would be advantageous to provide a method for computer system errors to be reliably reported to a manufacturer in such a manner that the process is automated to the level of determining whether the problem is known or unknown. And, if the problem is known, providing channels for delivery of updated software to clients, and if unknown, providing a method to obtain and analyze necessary information from the suspect system to enable diagnosis and correction of the errors.

Summary of the Invention

The invention includes a system and method for analyzing panic messages from computer systems that have suffered failures. One of the last processes a filer (server dedicated to file storage and retrieval) performs before it crashes is to render a panic message. This message is indicative of the problem that caused the filer to crash. This message is sent to the manufacturer via a communications network such as the Internet. The message also includes other information, such as the user's name, the version of the software, a back trace, and a mini core dump.

At the manufacturer, automatic analysis commences to determine if the bug can be identified. First, the panic message is analyzed by comparing it against a database of panic messages that correspond with known bugs. If successful, automated housekeeping occurs which includes updating this instance in a tracking database, delivery of an answer to the customer (including solutions), updating analysis statistics, and additional activities. If unsuccessful the process continues.

Second, a back trace analyzer analyzes the back trace using an expression algorithm that looks for exact matches on function names and recognized sequences of matches that correspond to known bugs. If successful, automated housekeeping occurs as indicated above. If unsuccessful, the process continues.

Third, a core script analyzer analyzes a core dump for recognizable patterns of code that correspond to known bugs. If successful, automated housekeeping occurs as indicated above If unsuccessful the process continues.

Fourth, if the automated methods above fail to identify the problem as a known bug, the core in manually analyzed to detect known and unknown bugs. Manual and automatic housekeeping is performed following this manual analysis.

Brief Description of the Drawings

Figure 1 illustrates a block diagram of a system for a panic message analyzer.

Figure 2 illustrates a panic message analyzer process in a system for a panic message analyzer.

Figure 3 illustrates an auto support process in a system for a panic message analyzer.

Figure 4 illustrates a core dump process in a system for a panic message analyzer.

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. Embodiment of the invention can be implemented using general purpose processors or special purpose processors operating under program control, or other circuits, adapted to particular process steps and data structures described herein. Implementation of the process steps and data structures described herein would not require undue experimentation or further investigation.

Lexicography

The following terms refer to or relate to aspects of the invention as described below. The descriptions of general meanings of these terms are not intended to be limiting, only illustrative.

- filer – This term refers to a file server. A file server is a computer and storage device dedicated to data storage and retrieval.

- Core dump - A core dump is the printing or the copying to a more permanent medium (such as a hard disk) the contents of random access memory at one moment in time.

- Mini core dump – A subset of data from a core dump.

- Back-trace – A list of computer instructions in the reverse order they were executed.

As noted above, these descriptions of general meanings of these terms are not intended to be limiting, only illustrative. Other and further applications of the invention, including extensions of these terms and concepts, would be clear to those of ordinary skill in the art after perusing this application. These other and further applications are part of the scope and spirit of the invention, and would be clear to those of ordinary skill in the art, without further invention or undue experimentation.

System Elements

Figure 1 shows a block diagram of a system for a panic message analyzer.

A system 100 includes a **client device 110** associated with a customer, a **communications link 120**, a **communications network 130**, a **server device 140** associated with a manufacturer, a **mass storage 150**, a **housekeeping database 151**, a **bugs database 152**, and a **core dump 160**.

The **client device 110** includes a processor, a main memory, and software for executing instructions (not shown, but understood by one skilled in the art). Although the **client device 110** and **server device 140** are shown as separate devices there is no requirement that they be separate devices.

The **communications link 120** operates to couple the **client device 110** to the **communications network** 130.

The **server device 140** includes a processor, a main memory, software for executing instructions (not shown, but understood by one skilled in the art), and a **mass storage 150**. Although the **client device 110** and **server device 140** are shown as separate devices there is no requirement that they be separate devices. Additionally, although the

server device 140 and **mass storage 150** are shown as combined there is no requirement that they be combined. They could be separate devices.

The **mass storage 150** includes the **housekeeping database 151** and **bugs database 152.**

The **core dump 160** includes a **mini core dump 161**, a **back-trace 162**, and a **panic message 163.**

Method of Operation – Manual Message Processing

Figure 2 illustrates a panic message analyzer process, indicated by general reference character 200. The manual panic message analyzer process 200 initiates at a 'start' terminal 201. The panic message analyzer process 200 continues to a 'panic message created' procedure 203 which allows the customer's device to create a **panic message 163** prior to failure.

A 'customer submits panic message' procedure 205 allows the customer to submit the **panic message 163** for analysis utilizing the **client device 110** to transmit the **panic message 163** to the **server device 140.** In a preferred embodiment, the customer submits the message via interaction and transfer over an Internet connection which is well-known in the art. There is, however, no requirement the **panic message 163** be transferred by this method as long as it is delivered to the manufacturer.

An 'analyze panic message' procedure 207 allows the **panic message 163** to be analyzed by comparing recognized data elements it contains (a panic message includes the address of where a system was last operating, line numbers, text and source code filenames, and other data) against known data elements that correspond to known bugs in the **bugs database 152** on the **server device 140.**

A 'known bug?' decision procedure 209 determines whether the panic message identifies a known bug. If the 'known bug?' decision procedure 209 determines

that the bug is a known bug, the panic message analyzer process 200 continues to a "solution to customer" procedure 213.

An 'investigate via auto support' procedure 211 takes note that analysis of the customer-submitted **panic message 163** failed to identify the problem with the affected system and that further investigation is required. The panic message analyzer process 200 continues to an "automatic housekeeping" procedure 215.

The 'solution to customer' procedure 213 extracts a solution from the database which is associated with the bug identified by the 'known bug' decision procedure 209. The solution provided to the customer can be written instructions detailing how to fix and avoid further occurrences, a copy of a software program to fix the problem, or recommendations for the purchase of additional products from the manufacturer that fix the problem.

An 'automatic housekeeping' procedure 215 records all relevant information regarding identification/non-identification of the bug, the solution sent to the customer (if any), and statistics relating to these events in the **housekeeping database 151**. If the panic message analyzer failed to diagnose the problem, the 'automatic housekeeping' procedure leaves the case active (i.e. marked as unresolved).

Method of Operation – Auto Support Analysis

Figure 3 illustrates an auto support process, indicated by general reference character 300. The auto support process 300 initiates at a 'start' terminal 301. The auto support process 300 continues to an 'auto support message sent' procedure 303 which allows the **client device 110** to automatically send a message to the **sever device 140** containing a copy of the **panic message 163** and **mini core dump 161**.

An 'auto support message received' procedure 305 allows the server device 140 to receive the **panic message 163** and **mini core dump 161** from the **client device 110**.

An 'analyze panic message' procedure 307 allows the **panic message 163** to be analyzed by comparing recognized data elements it contains (a panic message includes the address of where a system was last operating, line numbers, text and source code filenames, and other data) against known data elements that correspond to known bugs in the **bugs database 152** on the **server device 140.**

A 'known panic bug?' decision procedure 309 determines whether the panic message identifies a known bug. If the "known bug?' decision procedure 209 determines that the bug is a known bug, the panic message analyzer process 200 continues to a "discard mini core dump" procedure 321.

An 'extract back-trace' procedure 311 extracts the **back-trace 162** from the **mini core dump 161**.

An 'analyze back-trace' procedure 313 allows the **back-trace 162** to be analyzed using an expression algorithm that looks for exact matches on function names and recognized sequences of function names that correspond to known bugs in the **bugs database 152** on the **server device 140.**

A 'known back-trace bug?' decision procedure 315 determines whether the **back-trace 162** identifies a known bug. If the 'known back-trace bug?' decision procedure 315 determines that the bug is a known bug, the auto support process 300 continues to a "discard mini core dump" procedure 321.

A 'request core dump' 317 procedure notifies the customer that a **core dump 160** of the affected system is required. This notification includes all the instructions necessary to create the **core dump 160** and deliver it to the manufacturer. In a preferred embodiment, the notification would be sent electronically to the customer; however, there is no requirement that notification be accomplished in this manner.

An 'automatic housekeeping' procedure 319 records all relevant information regarding identification/non-identification of the bug, the solution sent to the customer (if any), and statistics relating to these events in the **housekeeping database 151**. If the panic

message analyzer failed to diagnose the problem, the 'automatic housekeeping' procedure leaves the case active (i.e. marked as unresolved).

Additionally, if the bug was identified by analysis of the back-trace, functionality exists that allows the back-trace analyzer to teach the panic message analyzer about the bug. This allows future instances of the bug to be resolved at an earlier stage.

For example, if the bug first appeared in version one of the software at line 10, the panic message analyzer would not identify it in version two if the bug now appeared at line 20 due to the exact matching methodology used. The back-trace analyzer, however, might identify the bug as it uses a more sophisticated approach, and it would then pass this information to the panic message analyzer. The auto support process 300 terminates through an 'end' terminal 325.

A 'discard mini core dump' procedure 321 causes the **mini core dump 161** to be discarded as it is no longer needed due to identification of the bug.

A 'solution sent to customer' procedure 323 causes a solution to be extracted from the **bugs database 152** which is associated with the identified bug. The solution provided to the customer varies depending on the bug identified. For example, it can be written instructions detailing how to fix and avoid further occurrences, a copy of a software program to fix the problem, or recommendations for the purchase of additional products from the manufacturer that fix the problem.

The auto support process 300 continues to an 'automatic housekeeping' procedure 319.

Method of Operation – Core Dump Analysis

Figure 4 illustrates a core dump process, indicated by general reference character 400. The core dump process 400 initiates at a 'start' terminal 401. The core dump process 400 continues to a 'core arrives from customer' procedure 403 which allows analysis

of the **core dump 160** to begin. The **core dump 160** is requested by a' request core dump' procedure 317 (illustrated in Figure 3) when prior analysis of the **panic message 163** and **back-trace 162** have failed. These two analysis techniques, however, are duplicated during the core dump process 400 further providing fail-safe systematic analysis.

An 'analyze panic message' procedure 405 allows the **panic message 163** to be analyzed by comparing recognized data elements it contains (a panic message includes the address of where a system was last operating, line numbers, text and source code filenames, and other data) against known data elements that correspond to known bugs in the **bugs database 152** on the **server device 140.**

A 'known panic bug?' decision procedure 407 determines whether the panic message identifies a known bug. If the "known bug?' decision procedure 407 determines that the bug is a known bug, the core dump process 400 continues to a "store core dump" procedure 423.

An 'extract back-trace' procedure 409 extracts the **back-trace 162** from the **core dump 160.**

An 'analyze back-trace' procedure 411 allows the **back-trace 162** to be analyzed using an expression algorithm that looks for exact matches on function names and recognized sequences of function names that correspond to known bugs within the **bugs database 152.**

A 'known back-trace bug?' decision procedure 413 determines whether the **back-trace 162** identifies a known bug. If the 'known back-trace bug?' decision procedure 413 determines that the bug is a known bug, the core dump process 400 continues to a 'store core dump' procedure 423.

A 'core script analyzer' procedure 415 automatically analyzes the **core dump 160** by searching for data elements in the **core dump 160** that correspond to known bugs within the **bugs database 152.**

A 'known core bug?' decision procedure 417 determines whether core script analysis has identified a known bug. If the 'known core bug?' decision procedure 417 determines it has identified a known core bug, the core dump process 400 continues to a 'store core dump' procedure 423.

A 'manual core dump analysis' procedure 419 allows the **core dump 160** to be analyzed manually by personnel at the manufacturer.

A 'manual solution sent to customer' procedure 421 allows personnel at the manufacturer to send a solution to the customer based on the manual analysis of the **core dump 160**. The core dump process 400 continues to a "automatic housekeeping" procedure 427.

A 'store core dump' procedure 423 allows the **mini core dump 161** to be moved to a storage location.

A 'solution sent to customer' procedure 425 causes a solution to be extracted from the **bugs database 152** which is associated with the identified bug. The solution provided to the customer varies depending on the bug identified. For example, it can be written instructions detailing how to fix and avoid further occurrences, a copy of a software program to fix the problem, or recommendations for the purchase of additional products from the manufacturer that fix the problem.

An 'automatic housekeeping' procedure 427 records all relevant information regarding identification/non-identification of the bug, the solution sent to the customer (if any), statistics relating to these events, and any entries necessary to the **bugs database 152**.

Additionally, if the bug was identified by analysis of the back-trace, functionality exists that allows the back-trace analyzer to teach the panic message analyzer about the bug. This allows future instances of the bug to be resolved at an earlier stage.

Furthermore, if the bug was identified by analysis of the core, functionality exists that allows the core to teach the back-trace analyzer and panic message analyzer about the bug. This allows future instances of the bug to be resolved at an earlier stage.

The core dump process 400 terminates through an 'end' terminal 429.

Generality of the Invention

The invention has general applicability to various fields of use, not necessarily related to the services described above. For example, these fields of use can include one or more of, or some combination of, the following:

- In addition to general applicability to file servers the invention has broad applicability to networks, network devices, and all types of software. Other and further applications of the invention, in its most general form, will be clear to those skilled in the art after perusal of this application, and are within the scope and spirit of the invention.

Alternate Embodiments

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Claims

1. A method of analyzing computer error messages comprising;

receiving a message including information regarding an extraordinary event comprising at least an address value, source code, and other information associated with said event; and

applying a set of rules to said message; and

generating an action in response to said message.

2. The method of claim 1, wherein **said other information** includes some text other than said address value.

3. The method of claim 1, wherein said message is responsive to an **operation** taken by a customer.

4. The method of claim 3, wherein said **operation** is electronic transfer of **said message** from **said customer** to a manufacturer via a communications network.

5. The method of claim 4, wherein said **electronic transfer** of **said message** includes a **product release version** as entered by **said customer**.

6. The method of claim 1, wherein **said applying a set of rules** compares said **message** against a **first database** of known bugs.

7. The method of claim 6, wherein **said applying a set of rules** further comprises identifying **said message** as matching a known bug in a **first database**.

8. The method of claim 7, wherein **said known bug** is linked to solutions associated with **said version**.

9. The method of claim 8, wherein a solution is extracted from **said solutions** by matching **said version** entered by **said customer** to **said version** associated with **said known bug.**

10. The method of claim 1, wherein said action is delivery of **said solution** to **said customer.**

11. The method of claim 10, wherein **said delivery** is by transmission over a communications network.

12. The method of claim 10, wherein **said delivery** is by a mail service.

13. The method of claim 1, wherein **said action** further comprises recording statistics to a second database.

14. The method of claim 13, wherein **said statistics** further comprise information related to said message.

15. A method of analyzing computer error messages comprising;

receiving a message including information comprising a sequence of executed program instructions;
applying a set of rules to said sequence; and
generating an action in response to said message.

16. The method of claim 15, wherein **said message** is responsive to a problem with a device utilized by **said customer** and manufactured by **said manufacturer.**

17. The method of claim 16, wherein **said message** further comprises an address value and a **product release version.**

18. The method of claim 17, wherein **said message** is electronically transferred via **said communications network.**

19. The method of claim 15 wherein **said message** is compared against a **first database** of known bugs.

20. The method of claim 19, wherein **said applying a set of rules** to said **sequence** comprises identifying some portion of **said sequence** as matching at least one of **said known bugs** in **said first database.**

21. The method of claim 20, wherein **said known bugs** are associated with solutions categorized by **said version.**

22. The method of claim 20, wherein a solution is extracted from **said solutions** by further matching **said version** received from **said customer** to **said version** associated with **said solutions.**

23. The method of claim 15, wherein **said action** is delivery of **said solution** to **said customer.**

24. The method of claim 23, wherein **said delivery** is by transmission over a communications network.

25. The method of claim 23, wherein **said delivery** is by a mail service.

26. The method of claim 15, wherein **said action** further comprises recording statistics to a second database.

27. The method of claim 26, wherein **said statistics** further comprise information related to said message.

28. The method of claim 26, wherein **said action** teaches, when possible, a prior identification process to recognize **said known bugs.**

29. A method for analyzing computer error messages comprising;

receiving a message including information comprising the entire contents of a random access memory of a device;

applying a set of rules to said contents; and

generating an action in response to said message.

30. The method of claim 29, wherein said message is responsive to a request by the manufacturer and to an initial problem with a device owned by **said customer** and manufactured by **said manufacturer.**

31. The method of claim 29, wherein **said contents** further comprises a sequence of executed program instructions, an address value, and a **product release version.**

32. The method of claim 29, wherein **said message** is electronically transferred via a **communications network.**

33. The method of claim 29, wherein **said applying a set of rules** further comprises scanning **said contents** for patterns of data that indicate known bugs.

34. The method of claim 33, wherein **said scanning** identifies some portion of **said contents** as matching at least one of **said known bugs** in **said first database.**

35. The method of claim 34, wherein **said known bugs** are associated with solutions categorized **by said version.**

36. The method of claim 35, wherein a solution is extracted from **said solutions** by matching **said version** received from **said customer** to **said version** associated with **said known bug.**

37. The method of claim 29, wherein **said action** is delivery of **said solution** to **said customer.**

38. The method of claim 37, wherein **said delivery** is by transmission over a network.

39. The method of claim 37, wherein **said delivery** is by a mail service.

40. The method of claim 29, wherein **said action** further comprises recording statistics to said second database.

41. The method of claim 40, wherein **said statistics** further comprise information related to said message.

42. The method of claim 29, wherein **said action** teaches, when possible, prior identification processes to recognize **said known bugs.**

43. Apparatus including a memory, said memory having storage capable of holding information, said information including;

information identifying a product release version;
information identifying a set of bugs in an earliest release; and
information regarding duplicate bugs.

44. The apparatus of claim 43, wherein **said version** is stored in a first database.

45. The apparatus of claim 43, wherein **said set of bugs** is stored in **said first database.**

46. The apparatus of claim 43, wherein **said version** and **said set of bugs** are associated.

47. The apparatus of claim 43, wherein **said duplicate bugs** are stored in **said first database.**

48. The apparatus of claim 47, wherein duplicate like bugs are removed from **said first database.**

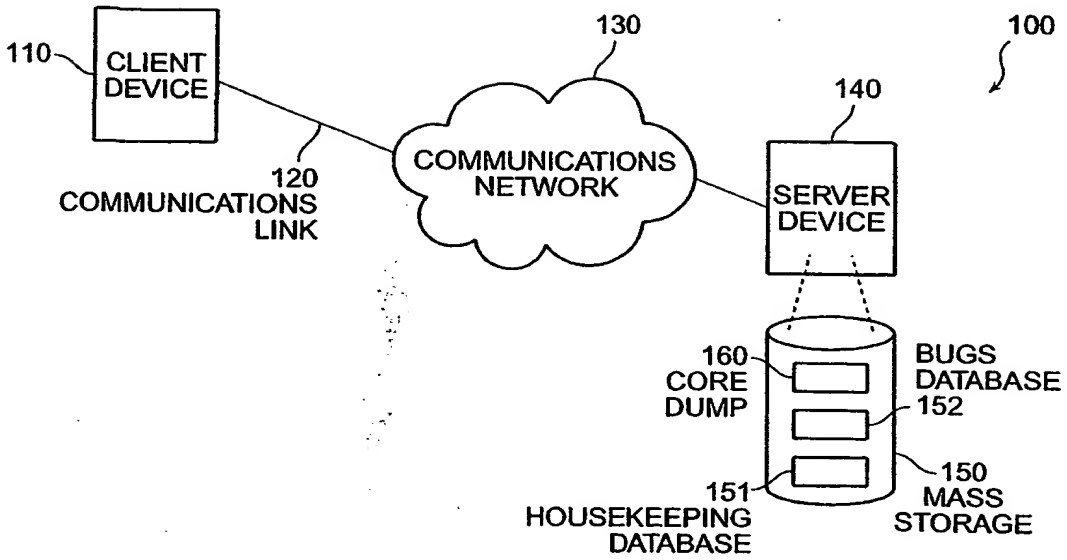

110
CLIENT DEVICE
120
COMMUNICATIONS LINK
130
COMMUNICATIONS NETWORK
140
SERVER DEVICE
100
160
CORE DUMP
BUGS DATABASE
152
150
MASS STORAGE
151
HOUSEKEEPING DATABASE

FIG. 1

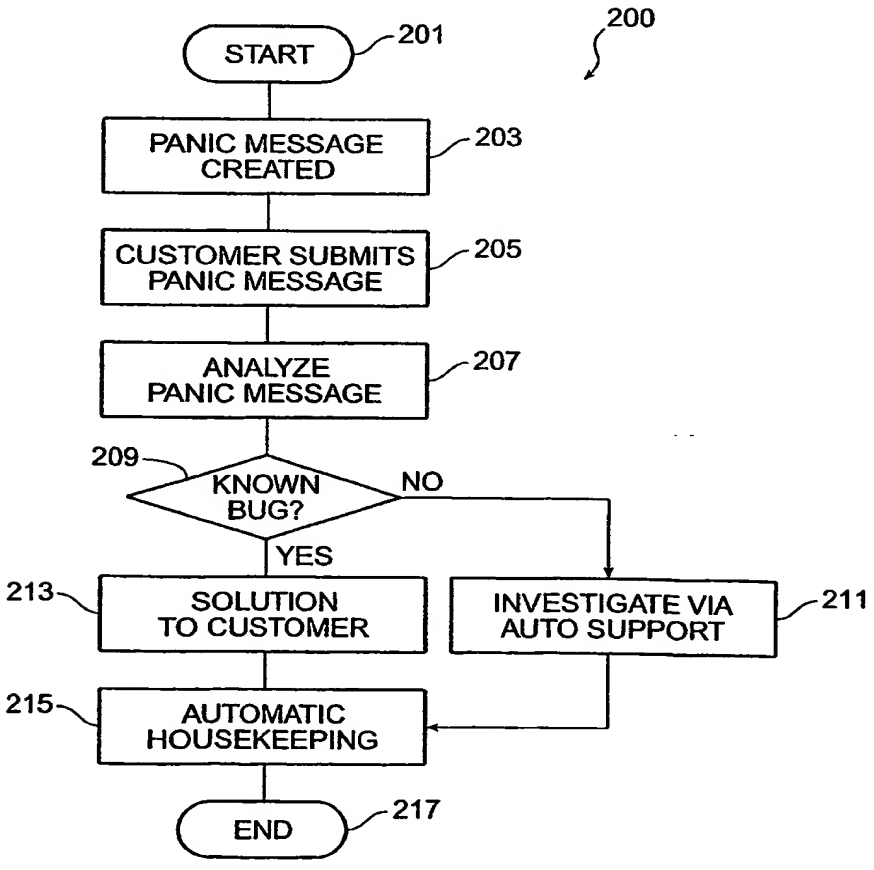

200
START
201
PANIC MESSAGE CREATED
203
CUSTOMER SUBMITS PANIC MESSAGE
205
ANALYZE PANIC MESSAGE
207
209
KNOWN BUG?
NO
YES
213
SOLUTION TO CUSTOMER
INVESTIGATE VIA AUTO SUPPORT
211
215
AUTOMATIC HOUSEKEEPING
END
217

FIG. 2

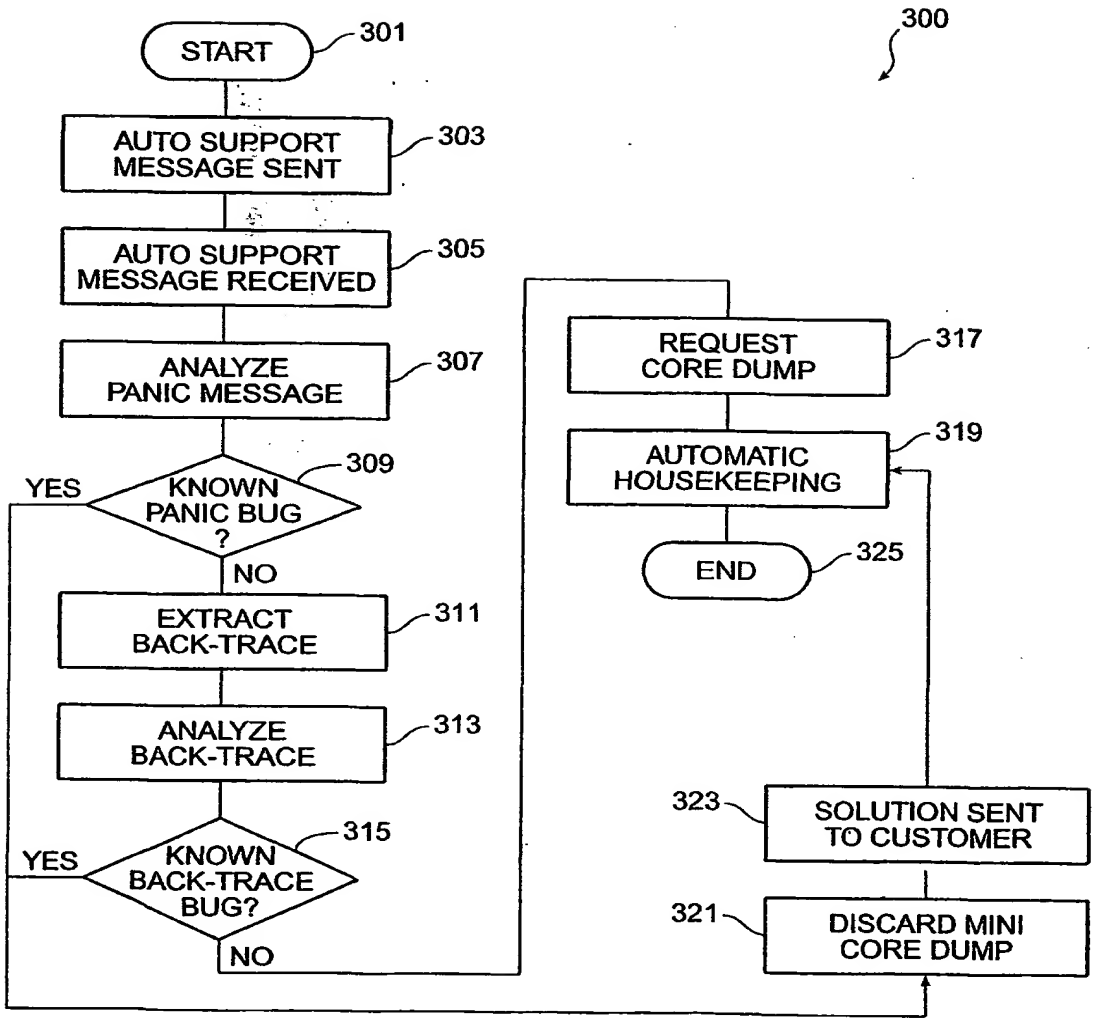

300
START
301
AUTO SUPPORT MESSAGE SENT
303
AUTO SUPPORT MESSAGE RECEIVED
305
ANALYZE PANIC MESSAGE
307
KNOWN PANIC BUG ?
309
YES
NO
EXTRACT BACK-TRACE
311
ANALYZE BACK-TRACE
313
KNOWN BACK-TRACE BUG?
315
YES
NO
REQUEST CORE DUMP
317
AUTOMATIC HOUSEKEEPING
319
END
325
SOLUTION SENT TO CUSTOMER
323
DISCARD MINI CORE DUMP
321

FIG. 3

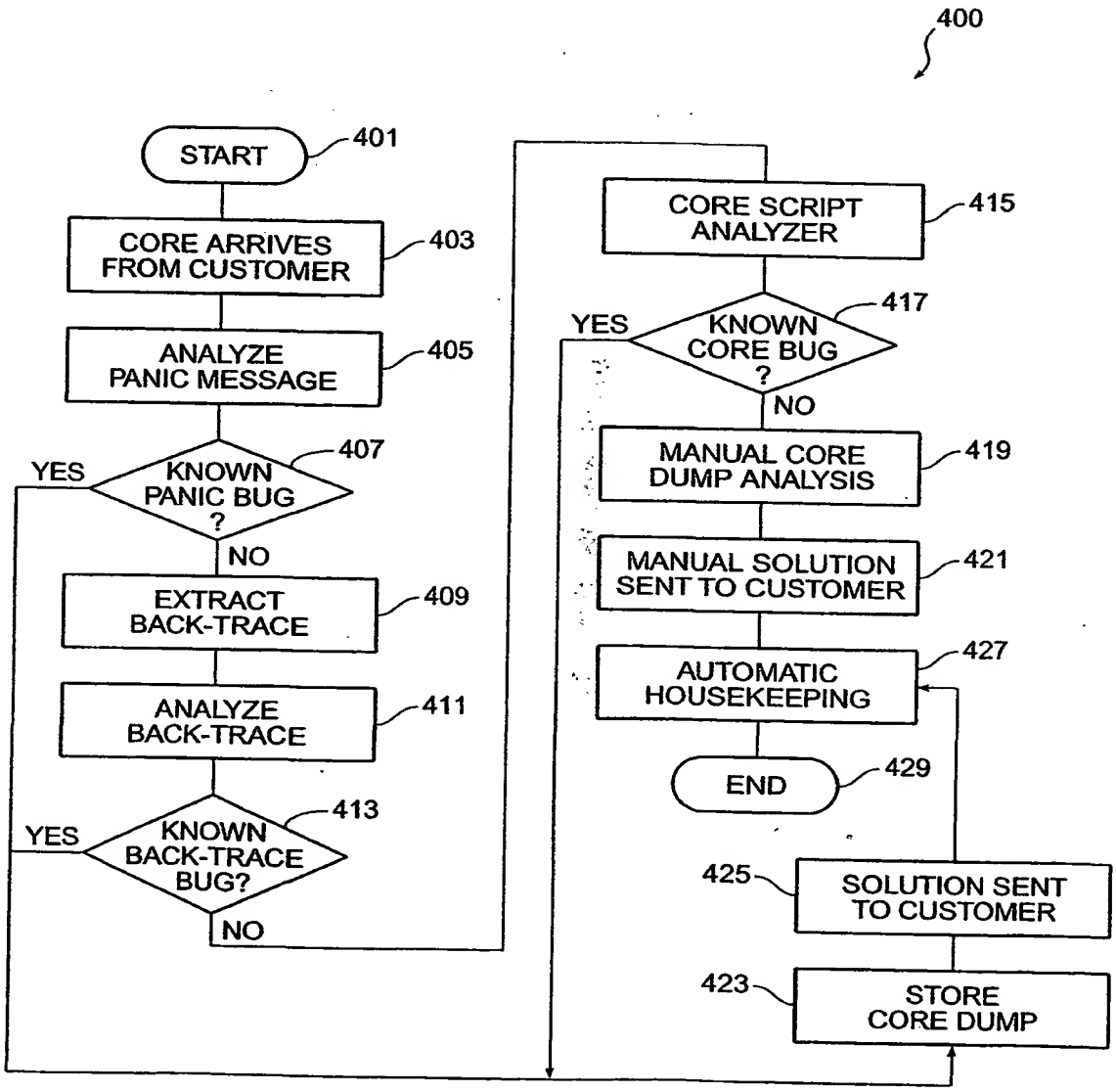

400
START
401
CORE ARRIVES FROM CUSTOMER
403
ANALYZE PANIC MESSAGE
405
KNOWN PANIC BUG ?
407
YES
NO
EXTRACT BACK-TRACE
409
ANALYZE BACK-TRACE
411
KNOWN BACK-TRACE BUG?
413
YES
NO
CORE SCRIPT ANALYZER
415
KNOWN CORE BUG ?
417
YES
NO
MANUAL CORE DUMP ANALYSIS
419
MANUAL SOLUTION SENT TO CUSTOMER
421
AUTOMATIC HOUSEKEEPING
427
END
429
SOLUTION SENT TO CUSTOMER
425
STORE CORE DUMP
423

FIG. 4